# EmailQueue

### Requirements

1. Oracle JDK. Here I have used version 1.8.0\_31.
2. Fake SMTP server that can be run on localhost using its jar file([fakeSMTP-1.13.jar](https://nilhcem.github.io/FakeSMTP/download.html) used in development). Default SMTP port (25) is used here. To start browse the folder and type **“sudo java-jar fakeSMTP-1.13.jar”.**
3. MySQL with Environment Path Set running on default port (3306).
4. JAVA mail API. Here, I have used “javax.mail.jar”.
5. MySQL JDBC connector. Here, I have used “mysql-connector-java-5.1.35-bin.jar”.

### Run the application

To run the application, browse to the folder and enter the following command.

“**java -jar /dist/Coupon\_Dunia.jar**”

Give the required inputs in the console to reconfigure the default constraints and settings.

### Classes and Methods used

### Classes:

* MainMail: Main class of the application.
* MultipleSend: Extends **Thread** class to run threads for sending emails.
* SetFieldsEmail: Class having attributes like 'to', 'from', 'sub', 'body' acting as a blueprint for emails.
* SqlEmail: Provides SQL related functions like FetchEmail, FetchDistinctEmail etc.
* SMTPSend: Enables to set SMTP constraints and send our required emails.

**Methods:**

* MainMail class
* main (): The main function.
* InputFromUser (): Shows the default configuration and if any reconfiguration is required it asks user to input new configuration parameters. Returns '1' or '0' based on which InsertDM () is called or not respectively.
* InsDM (): Calls **SetFieldsEmail ()** and sets 'to', 'from', 'sub' and 'body' fields and then calls SqlEmail.InsertMail () to insert dummy mails in the database. Here, a table is created in the database and dummy mails are put into it. Returns void.
* DistinctMailPairs (): Calls SqlEmail.FetchDistinctEmail () to fetch distinct

<to, from> email pairs from the database. It returns vector of String array if the execution is successful or throws an exception if not successful.

* SetFieldsEmail class
* SetFieldsEmail (): Constructor to create a new object that sets ‘to’, ‘from’, ‘sub’ and ‘body’ fields.
* save (): Calls SqlEmail.InsertMail() to save the SetFieldsEmail object as a new row in the database. Returns ’1’ or ‘0’ with respect to successful execution or not.
* MultipleSend class (extends Thread)
* MultipleSend(): Constructor to set ‘ThreadName’, ‘ThreadIndex’ and define vector of integers ‘sentId’.
  + run(): Starts the execution of the thread.
  + start(): Invokes the thread object to start its execution.
  + GetEmails(): Calls SqlEmail.FetchEmail() to get all the emails for a given ‘to’ and ‘from’ email pairs provided as arguments and whose sent\_bit = 0. Returns array of SetFieldsEmail objects having information of all the emails.
  + SendEmails(): Calls SMTPSend.SendEmail() to send all the mails for a given from and to email address. All the emails of a given to and from email address are passed as an array of EmailQueue objects. Returns true/false according to the execution is successful or not respectively.
* SqlEmail class
  + SqlEmail(): Constructor to create a new connection to MySQL server. Also, creates database and table if they does not exist.
  + InsertMail(): Inserts a new row in the desired table with fields (from\_email\_address, to\_email\_address, subject, body). Returns number of rows affected or -1 if the execution is unsuccessful.
  + FetchEmail(): Fetches all the emails whose sent\_bit = 0, corresponding to ‘from’ and ‘to’ fields. Returns vector of string array if the execution is successful or null if unsuccessful.
  + FetchDistinctEmail(): Fetches all the distinct <from, to> email pairs from the given table. Returns vector of string array if the execution is successful or null if unsuccessful.
  + MarkSentEmail(): Updates the sent\_bit to 1 if the emails are sent successfully. Returns number of rows updated for the successful execution or -1 for unsuccessful execution.
* SMTPSend class
  + SMTPSend(): Constructor to set properties and session object if not set previously. Authentication of password
  + SendEmail(): Sends all the emails provided as an array of SetFieldsEmail objects for a given ‘to’ and ‘from’ email address. Returns ‘1’ or ‘0’ with respect to successful execution or not respectively.

### Technique used

Steps

* 1. Main program extracts all the distinct pair of emails from our database created it stores the result in MainMail.mailPairs which is a vector of string arrays.
  2. Next, number of threads (MainMail.No\_of\_threads) are created with ThreadIndex starting from 1 to ‘MainMail.No\_of\_threads’ and their name (ThreadName) accordingly.
  3. Each and every thread is responsible to send emails of indices ‘ i+ MainMail.No\_of\_threads’ where ThreadIndex <= i <= size (mailPairs) – ThreadIndex.
  4. While thread sends the emails ‘id by id’, it updates their respective sent\_bit to 1, in order to prevent further sending of same mails again and again.